

# RAPPORT SUR LE BRIEF PROJET

## RESUME DU PROJET

DATE DU RAPPORT	NOM DU PROJET	AUTEUR
01/12/2019	Brief Projet – Scenario I	Ecoteam

## ÉTAT RECAPITULATIF

Dans ce scenario le but est de faire le tour sur les commandes les plus utilisées de l'outil GIT

## VUE D'ENSEMBLE DU PROJET

TACHE	NOTES
Immersion	Mettre en place le setup qui nous servira pour utiliser les commandes GIT
Découverte	Comprendre la configuration GIT
Historique	Pouvoir se repérer dans les commits
Excluding Files	Introduction au .gitignore
Branching and Merging	Créer des branches et pouvoir merger entre elle
Conflict Resolution	Comment lors du merging, pouvoir régler les conflicts entre contributeurs/collaborateurs
Merge Tools	Introduction au p4merge et son utilisation
Challenge	Pouvoir cibler avec le .gitignore les fichiers que nous devons conserver et ignorer le reste

## HISTORIQUE DES RISQUES ET PROBLEMES

PROBLEME	NOTES
Répartition des taches	La simplicité des tâches reste relatives quant à la difficulté de mettre 6 personnes sur ces dernieres
–	–
–	–

## CONCLUSIONS/RECOMMANDATIONS

Le début peut être difficile car personne ne savait ou se situer vraiment, suite à certaines initiatives, idées et avis nous avons pu répartir les tâches en trois partitions, kanban, rapport et git ainsi qu'un SM en tant que chef d'orchestre.

## VUE EN DETAIL DE L'ETAPE I – IMMERSION

ITEM	DESCRIPTION
Le GIT	GIT est un logiciel libre de gestion de versions sous licence GNU V2 le plus utilisé au monde
.git	Le fichier .git contient toutes les données relatives au GIT a besoin pour gérer l'historique
Explication du commentaire	Up-to-date with 'origin/master' a pour signification qu'il n'y a rien à pousser. 'Working directory clean' a pour signification que tous les fichiers du répertoire actuel sont gérés par GIT et que la version la plus récente du fichier a été validé
README.mb	L'utilisation des cmd 'touch, vim, add, commit, push' du README.mb nous permet d'intégrer ce dernier

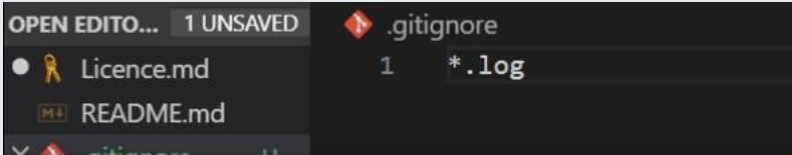
## VUE EN DETAIL DE L'ETAPE II - DECOUVERTE

ITEM	DESCRIPTION
HEAD	Une référence au dernier commit de la branche de sortie actuelle.
LOGS	Un enregistrement courant des commits présentant toutes les métadonnées relatif au log (auteur, adresse... )
BRANCHES	Un pointeur mobile sur l'un des commits effectués
Fichiers traqués	<pre>~/emacs.d(master) \$&gt; git status On branch master Changes to be committed:   (use "git reset HEAD &lt;file&gt;..." to unstage)      modified:   .gitignore     modified:   auto-save-list/.save-8600-nick-ThinkPad-X200~     modified:   projectile.cache</pre>

## VUE EN DETAIL DE L'ETAPE III - HISTORIQUE

ITEM	DESCRIPTION
Afficher le dernier commit avec option d'affichage	<pre>youcode@DESKTOP-SKHEMAK MINGW64 ~/Desktop/projects/demo (master) \$ git log --oneline --graph --decorate --all * f85c642 (HEAD -&gt; master) creat licence * 2ea9cbd readme is done</pre>
Créer son alias	'alias historique="git log -graph -decorate -oneline"
Historique des alias	<pre>youcode@DESKTOP-SKHEMAK MINGW64 ~/Desktop/projects/demo (master) \$ alias alias historique='git log --all --graph --decorate --oneline' alias ll='ls -l' alias ls='ls -F --color=auto --show-control-chars' alias node='winpty node.exe'</pre>
Historique des commit README	<pre>youcode@DESKTOP-SKHEMAK MINGW64 ~/Desktop/projects/demo (master) \$ historique readme.md * 2ea9cbd readme is done</pre>

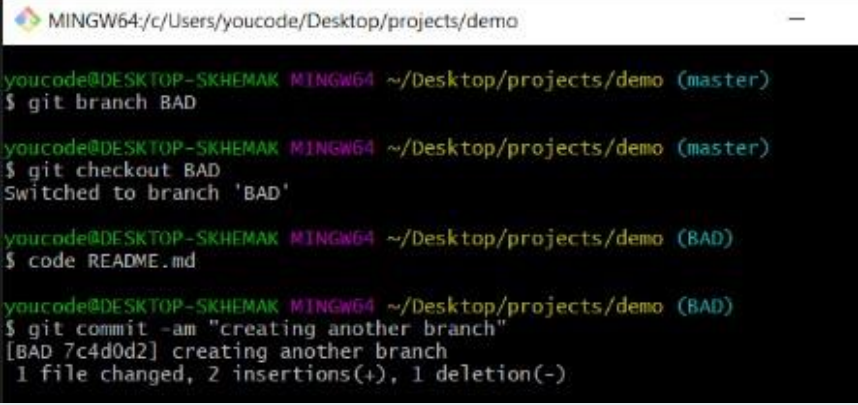
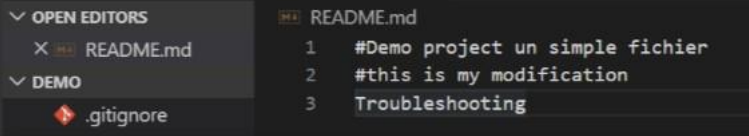
## VUE EN DETAIL DE L'ETAPE IV – EXCLUDING FILES

ITEM	DESCRIPTION
Stagging	<pre> youcode@DESKTOP-SKHEMAK MINGW64 ~/Desktop/projects/demo (master) \$ mv -n licence.md licence.txt  youcode@DESKTOP-SKHEMAK MINGW64 ~/Desktop/projects/demo (master) \$ git status On branch master Changes not staged for commit:   (use "git add/rm &lt;file&gt;..." to update what will be committed)   (use "git restore &lt;file&gt;..." to discard changes in working directory)         deleted:    Licence.md  Untracked files:   (use "git add &lt;file&gt;..." to include in what will be committed)         licence.txt  no changes added to commit (use "git add" and/or "git commit -a")  youcode@DESKTOP-SKHEMAK MINGW64 ~/Desktop/projects/demo (master) \$ git commit -am "licence changed from md to txt" [master 72cb42a] licence changed from md to txt 1 file changed, 0 insertions(+), 0 deletions(-) delete mode 100644 Licence.md </pre>
.gitignore	 <pre> youcode@DESKTOP-SKHEMAK MINGW64 ~/Desktop/projects/demo (master) \$ touch application.log  youcode@DESKTOP-SKHEMAK MINGW64 ~/Desktop/projects/demo (master) \$ touch .gitignore  youcode@DESKTOP-SKHEMAK MINGW64 ~/Desktop/projects/demo (master) \$ code . ./          ../          .git/       .gitignore  youcode@DESKTOP-SKHEMAK MINGW64 ~/Desktop/projects/demo (master) \$ code .gitignore  youcode@DESKTOP-SKHEMAK MINGW64 ~/Desktop/projects/demo (master) \$   </pre>
Constation	Nous constatons que le fichier .log n'a pas été compris lors du stagging

## VUE EN DETAIL DE L'ETAPE V – BRANCHING & MERGING

ITEM	DESCRIPTION
The fast forward merging	Une fusion rapide peut se produire lorsqu'il existe un chemin linéaire entre le bout de la branche actuelle et la branche cible. Au lieu de la branche actuelle et la branche cible. Au lieu de « fusionner » réellement les branches, tout ce que Git doit faire pour intégrer les historiques et de déplacer (c'est-à-dire, « avancer rapidement ») de la branche actuelle jusqu'à la branche cible.
The manual	Une fois que les étapes de « préparation à la fusion » décrites précédemment ont été effectuées, une fusion peut être lancée en exécutant git merge <nom de la branche> et le nom de la branche qui sera fusionnée dans la branche réceptrice
The automatic	La fusion Git combine des séquences de commits dans un historique unifié des commits. Il y a deux manières principales de fusionner Git : Fast Forward et Three way. Git peut fusionner automatiquement les commits, sauf si des modifications entrent en conflit dans les deux séquences de validation.
Constation	—

VUE EN DETAIL DE L'ETAPE VI – CONFLICT RESOLUTION

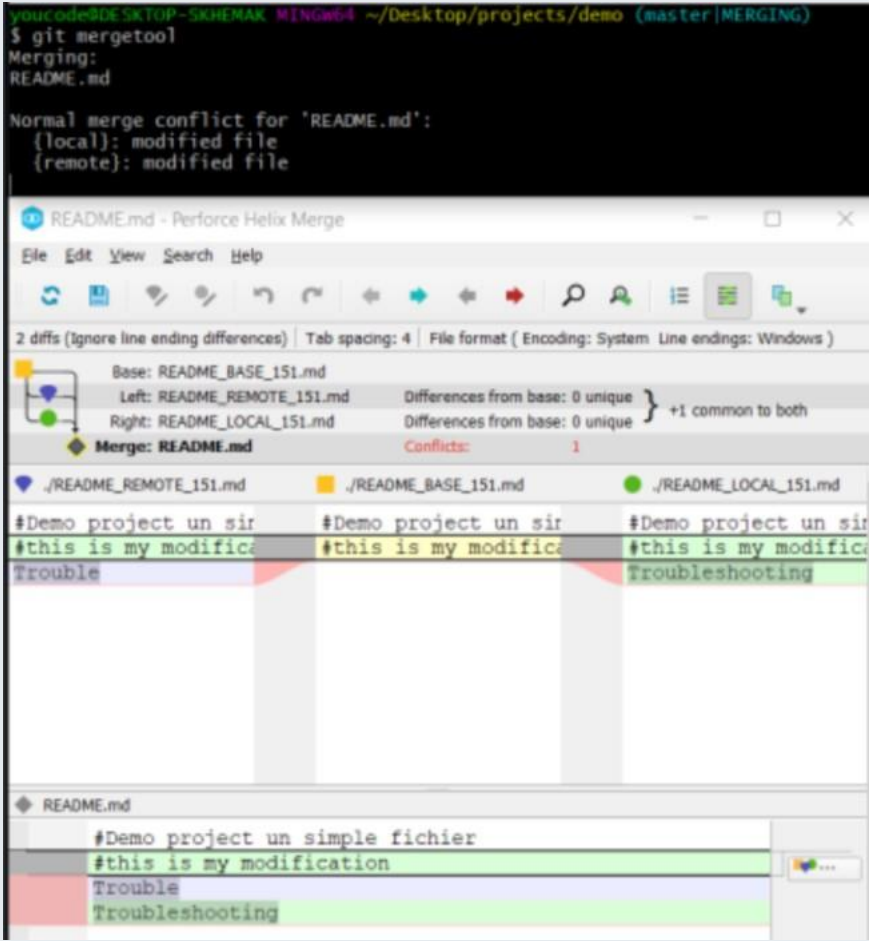
ITEM	DESCRIPTION
La ligne 'Trouble'	<div><pre>C: &gt; Users &gt; youcode &gt; Desktop &gt; projects &gt; demo &gt; README.md 1  #Demo project un simple fichier 2  #this is my modification 3  Trouble</pre><pre>youcode@DESKTOP-SKHEMAK MINGW64 ~/Desktop/projects/demo (master) \$ git branch BAD  youcode@DESKTOP-SKHEMAK MINGW64 ~/Desktop/projects/demo (master) \$ git checkout BAD Switched to branch 'BAD'  youcode@DESKTOP-SKHEMAK MINGW64 ~/Desktop/projects/demo (BAD) \$ code README.md  youcode@DESKTOP-SKHEMAK MINGW64 ~/Desktop/projects/demo (BAD) \$ git commit -am "creating another branch" [BAD 7c4d0d2] creating another branch 1 file changed, 2 insertions(+), 1 deletion(-)</pre></div>
Correction du commentaire	<div><pre>youcode@DESKTOP-SKHEMAK MINGW64 ~/Desktop/projects/demo (BAD) \$ git switch master Switched to branch 'master'  youcode@DESKTOP-SKHEMAK MINGW64 ~/Desktop/projects/demo (master) \$ code README.md  youcode@DESKTOP-SKHEMAK MINGW64 ~/Desktop/projects/demo (master) \$ git commit -am "branche bien faite" [master eb26611] branche bien faite 1 file changed, 2 insertions(+), 1 deletion(-)  youcode@DESKTOP-SKHEMAK MINGW64 ~/Desktop/projects/demo (master) \$ git commit --amend hint: Waiting for your editor to close the file... [master 49ec285] branche bien faite Date: Tue Nov 26 20:38:21 2019 +0100 1 file changed, 2 insertions(+), 1 deletion(-)  youcode@DESKTOP-SKHEMAK MINGW64 ~/Desktop/projects/demo (master) \$ historique bash: historique: command not found  youcode@DESKTOP-SKHEMAK MINGW64 ~/Desktop/projects/demo (master) \$ alias alias ll='ls -l' alias ls='ls -F --color=auto --show-control-chars' alias node='winpty node.exe'</pre></div>
Explication	Un conflit peut arriver si une deuxième modification coïncide avec les mêmes lignes de code lors du merge
Constation	Nous constatons que le p4merge ne pas disponible dans le chemin proposé par défaut

VUE EN DETAIL DE L'ETAPE VII – MERGE TOOLS

ITEM	DESCRIPTION
Configuration du p4merge	<pre>youcode@DESKTOP-SKHEMAK MINGW64 ~/Desktop/projects/demo (master MERGING) \$ git config --global --list core.editor="C:\Users\youcode\AppData\Local\Programs\Microsoft VS Code\Code.exe" --wait user.name=AYB-cmd user.name=AYB-cmd user.email=aeljelladi@gmail.com diff.tool=p4merge difftool.p4merge.path=C:\Program Files\Perforce\p4merge.exe mergetool.keepbackup=false mergetool.prompt=false mergetool.p4merge.path=C:\Program Files\Perforce\p4merge.exe merge.tool=p4merge alias.historique=git log --oneline --graph --decorate --all  youcode@DESKTOP-SKHEMAK MINGW64 ~/Desktop/projects/demo (master MERGING) \$ git config --global merge.tool p4merge  youcode@DESKTOP-SKHEMAK MINGW64 ~/Desktop/projects/demo (master MERGING) \$ git config --global mergetool.p4merge.path '/c/Program Files/Perforce/p4.exe'  youcode@DESKTOP-SKHEMAK MINGW64 ~/Desktop/projects/demo (master MERGING) \$ git config --global mergetool.prompt false  youcode@DESKTOP-SKHEMAK MINGW64 ~/Desktop/projects/demo (master MERGING) \$ git config --global diff.tool p4merge  youcode@DESKTOP-SKHEMAK MINGW64 ~/Desktop/projects/demo (master MERGING) \$ git config --global difftool.p4merge.path '/c/Program Files/Perforce/p4.exe'  youcode@DESKTOP-SKHEMAK MINGW64 ~/Desktop/projects/demo (master MERGING) \$ git config --global mergetool.prompt false  youcode@DESKTOP-SKHEMAK MINGW64 ~/Desktop/projects/demo (master MERGING) \$ git config --global --list core.editor="C:\Users\youcode\AppData\Local\Programs\Microsoft VS Code\Code.exe" --wait user.name=AYB-cmd user.name=AYB-cmd user.email=aeljelladi@gmail.com diff.tool=p4merge difftool.p4merge.path=C:/Program Files/Perforce/p4.exe mergetool.keepbackup=false mergetool.prompt=false mergetool.p4merge.path=C:/Program Files/Perforce/p4.exe merge.tool=p4merge alias.historique=git log --oneline --graph --decorate --all</pre>
Analyse	Il semblerait que p4merge fait une comparaison des deux modifications effectué dans la même ligne de code et nous permet de choisir celle qui nous convient et ainsi régler le conflit



Capture d'écran



VUE EN DETAIL DE L'ETAPE VIII – CHALLENGE

ITEM	DESCRIPTION
Screenshot	The screenshot shows the Visual Studio Code interface. The Explorer pane on the left shows the file structure with 'README.md' and '.gitignore'. The Source Control pane shows the current branch 'master' and untracked files. The main editor area shows the content of the '.gitignore' file, which includes 'README_*.md' and '*.log'. A terminal window is open at the bottom, showing the commands 'code .gitignore', 'git status', and the output indicating that the files are untracked and need to be added to the commit.